Amendments to the Drawings:

Figures 5A-C have been amended herewith to include the legend --PRIOR ART--.

Attachments: 1 Replacement Sheet

1 Annotated Sheet Showing Changes

REMARKS/ARGUMENTS

Claims 2-22 are pending in the present application. Claim 1 was canceled. Claims 2, 6-7, 12, 16-17 and 22 were amended. Reconsideration of the claims is respectfully requested.

I. Objection to Drawings

The Examiner objected to the drawings, stating that Figures 1-4 and 5A-C should be designated by a legend such as --Prior Art--.

In response to this objection, the legend --PRIOR ART-- has been applied to each of the Figures 5A-C.

In regard to Figures 1-4, Applicants clearly teach, such as at the Brief Description of the Drawings of their specification, that Figures 1 and 2 each shows a data processing system in which Applicants' invention may be implemented. Figure 3 shows software components within a data processing system that may implement the invention. In using the configurations respectively shown by Figures 1-3 to implement embodiments of Applicants' invention, such configurations are made unique to Applicants' teachings, and are therefore definitely not prior art. Figure 4 states specifically that the JVM shown thereby is in accordance with an embodiment of the invention, and thus is likewise not prior art.

For the above reasons , the objection to the drawings has been overcome.

II. Rejection of Claims 2-6 and 12-16 under 35 U.S.C. § 101

The Examiner has rejected Claims 2-6, 12-16 and 22 under 35 U.S.C. § 101 as being directed toward non-statutory subject matter, for not providing a useful, concrete and tangible result.

In response to this rejection, Claims 2-6 and 12-16 have been amended to recite the useful, concrete and tangible result of using a plug-in class loader that is provided for and delegated to an identified class loader to load a plug-in class that is associated with a specified application class. Accordingly, this rejection has been overcome.

III. Rejection of Claim 22 under 35 U.S.C. § 101

The Examiner has rejected Claim 22 under 35 U.S.C. § 101 as being not limited to tangible embodiments. This rejection is respectfully traversed.

The Examiner asserts that Claim 22 is not limited to tangible embodiments, and is therefore non-statutory, because such claim is directed to a medium that includes intangible embodiments, e.g. transmission-type media, such as radio frequency and light wave transmissions. However, no basis is present for holding a computer usable medium claim non-statutory because the medium may be allegedly "intangible." The MPEP states:

> In this context, "functional descriptive material" consists of **data structures** and computer programs **which impart functionality when employed as a computer component**. (The definition of "data structure" is "a physical or logical relationship among data elements, designed to support specific data manipulation functions." The New IEEE Standard Dictionary of Electrical and Electronics Terms 308 (5th ed. 1993).) "Nonfunctional descriptive material" includes but is not limited to music, literary works and a compilation or mere arrangement of data.
>
> **When functional descriptive material is recorded on some computer-readable medium it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized.** Compare *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994) (claim to data structure stored on a computer readable medium that increases computer efficiency held statutory) and *Warmerdam*, 33 F.3d at 1360-61, 31 USPQ2d at 1759 (claim to computer having a specific data structure stored in memory held statutory product-by-process claim) with *Warmerdam*, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure *per se* held nonstatutory). **(emphasis added) MPEP 2106 (IV)(B)(1)**

Claim 22 recites clearly functional descriptive material, since such material imparts functionality when employed as a computer component. Moreover, the functional descriptive material of Claim 22 is recorded on "some" computer readable medium.

In the above context, the term "some" means "any" computer readable medium. The MPEP does not draw any distinctions between one type of media that is considered to be statutory and another type of media that is considered to be non-statutory. To the contrary, the MPEP clearly states that as long as the functional descriptive material is in "some" computer readable medium, it should be considered statutory. The only exception to this statement in the MPEP is functional descriptive material that does not generate a useful, concrete and tangible result, e.g., functional descriptive material composed completely of pure mathematical concepts that provide no practical result. Claim 22 clearly recites a useful, concrete and tangible result by using a plug-in class loader that is provided for and delegated to an identified class loader in order to load a plug-in class that is associated with a specified application class.

Accordingly, Claim 22 is directed to functional descriptive material that provides a useful, concrete and tangible result, and which is embodied on "some" computer readable medium. Therefore, Claim 22 is statutory, and the rejection thereof under 35 U.S.C. § 101 has been overcome.

Moreover, use of a computer readable medium such as a floppy disc or CD-ROM in Claim 22 would clearly raise no issue of non-statutory subject matter. Devices of these types are characterized by the ability to store computer-related data for periods of time, and data can be written thereinto and be read therefrom at will. Applicant considers that if desired, those of ordinary skill in the art could readily conceptualize and construct a computer readable medium that had all the above characteristics of a CD-ROM or the like, and at the same time used one or more transmission-type or wireless communication links as a primary storage element. For example, the communication links could be connected to form a closed loop containing data packets in the form of electromagnetic energy. The packets would be continually circulated around the loop for a specified period of time, and data could be read therefrom and written thereinto.

IV. <u>35 U.S.C. § 112, Second Paragraph</u>

The Examiner has rejected Claim 1 under 35 U.S.C. § 112, second paragraph, as being indefinite for having insufficient basis for "the application class loader hierarchy".

Claim 1 has now been canceled. Accordingly, this rejection under 35 U.S.C. § 112, second paragraph has been overcome.

V. <u>35 U.S.C. § 102, Anticipation</u>

The Examiner has rejected Claims 2, 7-12 and 17-22 under 35 U.S.C. § 102 as being anticipated by U.S. Application Publication No. 2002/0184226 to Klicnik et al (hereinafter "Klicnik"). This rejection is respectfully traversed.

VI. <u>35 U.S.C. § 103, Obviousness</u>

The Examiner has rejected Claims 3-6 and 13-16 under 35 U.S.C. § 103 as being unpatentable over Klicnik in view of teachings related to Figure 5B of the application, which Applicants provided to illustrate a problem of the prior art. This rejection is respectfully traversed.

VII. <u>Teachings of Applicants</u>

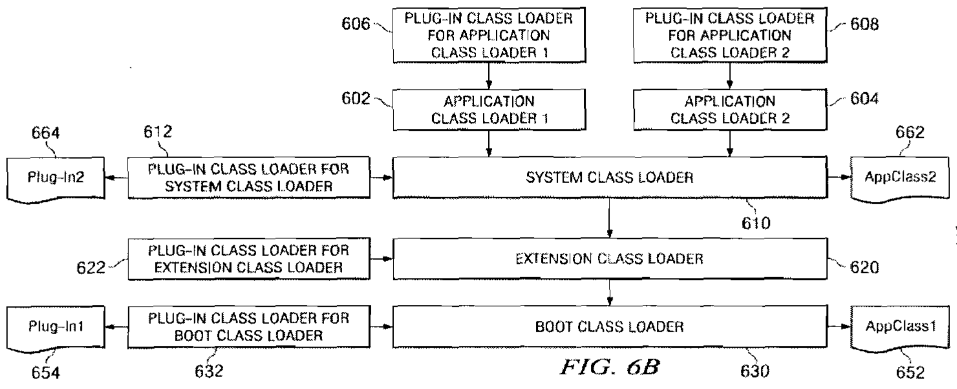
In their invention, Applicants provide a plug-in loader for each existing application class loader associated with a class loader hierarchy. In addition, a different plug-in class loader is provided for each class loader in the class loader hierarchy, and each plug-in class delegates to its corresponding class loader. The class loader used to load a particular application class is identified, and the plug-in class loader corresponding to the identified class loader is then used to load any plug-in class that is associated with the particular application class. Embodiments of the invention make it unnecessary to modify class path in order to load plug-in classes, and also provide other benefits and advantages as described in the application.

These teachings, set forth hereinafter, are found in the specification such as at page 18, lines 25 - page 19, line 2; page 19, line 24 - page 20, line 4; page 20, lines 11-20; and at Figure 6B.

> To solve the above mentioned problems, the present invention provides a set of plug-in class loaders in the class loader hierarchy. A plug-in class loader is provided for each class loader in the class loader hierarchy. Each plug-in class loader is associated with a single application class loader and is configured such that it delegates to its associated application class loader. **[page 18, line 25 – page 19, line 2]**

As stated above, the present invention provides a plug-in class loader for each existing application class loader. Plug-in class loader **606** is provided for and delegates to application class loader 1 **602**. Plug-in class loader **608** is provided for and delegates to application class loader 2 **604**. Similarly, plug-in class loader **612** is provided for and delegates to system class loader **610**; plug-in class loader **622** is provided for and delegates to extension class loader **620**; and, plug-in class loader **632** is provided for and delegates to boot class loader **630**. **[page 19, line 24 – page 20, line 4]**

Figure 6B illustrates the loading of plug-in classes by application classes using the loading class hierarchy of the present invention. The application class AppClass1 **652** is loaded by the boot class loader. Therefore, when the application class loads plug-in class Plug-In1 **654**, AppClass1 uses the plug-in class loader that delegates to the boot class loader to initiate the loading of Plug-In1. Similarly, since the application class AppClass2 **662** is loaded by the system class loader, AppClas2 uses the plug-in class loader that delegates to the system class loader to initiate the loading of the plug-in class, Plug-In2 **664**.**[page 20 lines 11-22]**


606
PLUG-IN CLASS LOADER FOR APPLICATION CLASS LOADER 1
PLUG-IN CLASS LOADER FOR APPLICATION CLASS LOADER 2
608
602
APPLICATION CLASS LOADER 1
APPLICATION CLASS LOADER 2
604
664
612
662
Plug-In2
PLUG-IN CLASS LOADER FOR SYSTEM CLASS LOADER
SYSTEM CLASS LOADER
AppClass2
610
622
PLUG-IN CLASS LOADER FOR EXTENSION CLASS LOADER
EXTENSION CLASS LOADER
620
Plug-In1
PLUG-IN CLASS LOADER FOR BOOT CLASS LOADER
BOOT CLASS LOADER
AppClass1
654
632
630
652
FIG. 6B

VIII. Rejection of Claim 2

Applicants' Claim 2 now reads as follows:

2. A method for selecting a class loader to load a plug-in class within a class loader heirarchy, the method comprising the steps of:

generating a class loader hierarchy comprising a plurality of class loaders that includes two or more application class loaders, one for each of two or more application classes, wherein each of said application class loaders can selectively load its application class, or delegate the loading of its application class to another class loader of said class loader hierarchy;

providing a different plug-in class loader for each class loader in the class loader hierarchy, wherein each plug-in class loader delegates to its corresponding class loader;

identifying the class loader of said hierarchy that is used to load a specified one of said two or more application classes; and

using the plug-in class loader that is provided for and delegates to said identified class loader to load a plug-in class that is associated with a said specified application class.

In rejecting Claim 2 as being anticipated by Klicnik, the Examiner stated the following:

13. Regarding Claims 2 and 12: Klicnik discloses a method for selecting a class loader for a plug-in, the method comprising:

providing a class loader hierarchy, wherein the class loader hierarchy includes a plurality of class loaders (Fig. 2); and

providing a plug-in class loader for each class loader in the class loader hierarchy (par. [0031] "each plug-in must be provided with a dedicated class loader"), wherein each plug-in class loader delegates to a respective class loader (par. [0043] "the class loader for parent is checked next"). **[Office Action dated April 9, 2007, page 5]**

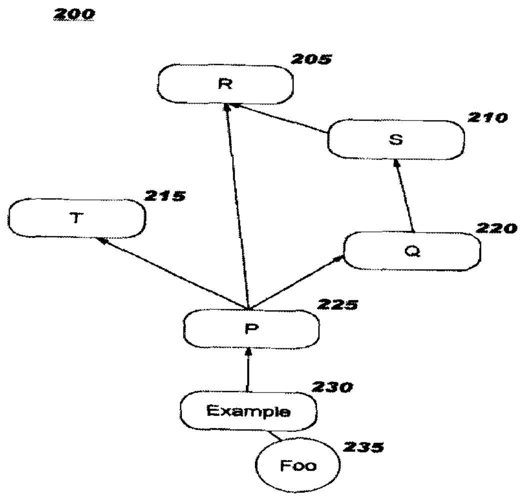
Pertinent teachings of Klicnik are found at paragraphs [0031], [0032] and [0043], and at Figure 2 thereof. These sections are as follows:

[0031] To maintain security, configurability, and scalability, each plug-in must be provided with a dedicated class loader when using the present invention. As in the prior art, the immediate class path of this class loader contains only the code (i.e. the jars and the directories) for the related plug-in. Since the code in a plug-in may refer to code provided in other plug-ins, a complex prerequisite system of functional dependencies among the plug-ins may be formed. A relatively simple example is depicted in FIG. 2. In this example prerequisite system 200, a plug-in named "Example" 230 contains the class Foo 235. The Example plug-in is dependent on code provided by plug-in P 225. (The direction of the arrows in FIG. 2 indicates the order of each dependency or prerequisite relationship.) Plug-in P, in this example, is dependent on plug-in R 205, plug-in T 215, and plug-in Q 220. Plug-in Q is dependent on plug-in S 210, which in turn is dependent on plug-in R 205. As will be obvious, the graph in this prerequisite system 200 does not adhere to the single inheritance model supported by the prior art dynamic class loaders, which has been described with reference to the tree structure 110 in FIG. 1B. (It should be noted that the terms "plug-in" and "component" are used interchangeably herein to describe the features of the present invention.)

[0032] According to the present invention, a particular plug-in specifies its prerequisites explicitly. For example, plug-in P explicitly states that it requires plug-in Q if P needs to refer to classes in Q during execution. FIG. 3 illustrates an example 300 of specifying this information, where the relationships in the prerequisite system depicted in FIG. 2 have been specified for each of the six plug-ins which were illustrated therein (see elements 330, 340, 350, 360, 370, and 380), as well as for the "Parent" 310 and "Platform" 320 plug-ins. (All plug-ins implicitly have the Platform as a prerequisite and all class loaders have Parent as their parent.) So, for example, the specification 340 for plug-in P indicates that P is dependent upon (i.e. has as prerequisites) code in R and Q. (As will be obvious, the information illustrated in FIG. 3 is merely representative of one manner in which the information used by the present invention may be specified.)

[0043] Referring now to FIG. 5, several example searches are shown which are based on the prerequisite structure in FIG. 2 and the plug-in specifications in FIG. 3. Assume that the class Foo is defined in the plug-in Example, as shown in FIG. 2, and that the syntax "new Bar ( )" is evaluated in a method of class Foo. The leftmost plug-in name in each example of FIG. 5, which is shown enclosed in parentheses, indicates where the class Bar is actually defined for each example search. In example 510, class Bar is located in the Parent plug-in. Thus, the search proceeds by first checking the cache of the class loader for the Example plug-in. Not finding Bar there, the class loader for Parent is checked next; Bar is found there, and the search ends successfully.

FIG. 2



A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Moreover, it is a fundamental principle of patent law that prior art must be considered in its entirety. **MPEP 2141.02.**

Applicants respectfully submit that *Klicnik* fails to teach every element of he claimed invention arranged as they are in Claim 1. For example, *Klicnik* does not teach, in the overall combination of Claim 1, the following Claim 1 features:

(1) Providing a different plug-in class loader for each class loader in the class loader hierarchy, wherein each plug-in class loader delegates to its corresponding class loader (hereinafter "Feature (1)").

(2) Identifying the class loader of the hierarchy that is used to load a specified one of the two or more application classes (hereinafter "Feature (2)").

(3) Using the plug-in class loader that is provided for and delegates to the identified class loader to load a plug-in class that is associated with the specified application class (hereinafter "Feature (3)").

IX. Claim 2 Distinguishes over Cited Reference

Feature (1) of Applicants' Claim 2 is directed to providing a different plug-in class loader for each class loader in the class loader hierarchy, wherein each plug-in class loader delegates to its corresponding class loader. This feature is disclosed in Applicants' specification, such as at page 20, lines 11-22, taken together with Figure 6B of Applicants' drawings. Figure 6B shows a class loader hierarchy comprising the class loaders 610, 620 and 630, and further comprising class loaders 602 and 604 for application classes 1 and 2, respectively. Moreover, Figure 6B shows each of the plug-in loaders 606, 608, 612, 622 and 632, wherein each class loader is provided with a different plug-in class loader.

In contrast to the recitation, of Feature (1), Klicnik is concerned with a technique for dynamically loading components with multiple prerequisite relationships, which are more complex than simple single inheritance chains. Paragraph [0031] emphasizes this purpose of Klicnik, by teaching a complex prerequisite system of functional dependencies among plug-ins. This teaching is further emphasized by Figure 2 thereof, which pictorially represents the dependency of a plug-in P on other plug-ins, such as R, T and Q. Also, paragraph [0032] teaches explicitly that a plug-in P requires a plug-in Q, if P needs to refer to classes in Q during execution.

Clearly, Figure 2 of Klicnik does not disclose a class loader hierarchy, as required by Feature (1) of Claim 2. Moreover, paragraph [0031] teaches only that each plug-in must be provided with a dedicated class loader for the invention thereof. Neither at such paragraph, nor anywhere else in Klicnik, is it taught or suggested to provide a different plug-in class loader for each class loader in a class loader hierarchy, as is required by Feature (1) of Claim 2. Klicnik likewise fails to disclose or suggest the Feature (1) requirement that each plug-in class loader delegates to its corresponding class loader.

Feature (2) of Claim 2 recites identifying the class loader of the hierarchy that is used to load a specified one of two or more application classes. Klicnik, however, appears to provide no teaching in regard to application classes, or to loading a specified application class. Klicnik certainly fails to teach the above Feature (2) requirement. Moreover, it is readily apparent that such requirement would have no purpose or be of any benefit in the scheme of Klicnik. Paragraph [0043] of Klicnik, cited in the Office Action in regard to original Claim 2, is concerned with searching for a plug-in, not with identifying a class loader of a hierarchy used to load a specified application class.

Feature (3) of Claim 2 recites using the plug-in class loader that is provided for and delegates to the identified class loader to load the plug-in class that is associated with the application class. Klicnik also fails to teach this feature. For example, in paragraph [0035] Klicnik states that a plug-in's class loader is used to load all of its classes. However, there is no disclosure or suggestion of using a plug-in class that is provided for the identified class of Claim 2, and delegated thereto, in order to load a plug-in class associated with the specified application class of Claim 2.

X. Remaining Claims Distinguished over Cited Reference

Claims 12 and 22 have respectively been amended to incorporate the subject matter of Features (1), (2) and (3) of Claim 2, and are each considered to distinguish over the art for the same reasons given in support thereof.

Claims 3-11 and 13-21 depend from Claims 2 and 12, respectively, and are each considered to distinguish over the art for at least the same reasons given in support thereof.

Claims 8 and 18 are additionally considered to distinguish over the art in reciting the steps of, responsive to a first application class load and a first plug-in class, identifying a target class loader within the class loader hierarchy that loaded a target class; identifying a plug-in class loader that is provided for and delegates to the target class loader; and loading the first plug-in class using the plug-in class loader. The cited Klicnik reference neither discloses nor suggests this feature.

Claims 9 and 19 are additionally considered to distinguish over the art in reciting that the step of identifying a target class loader within the class loader hierarchy that loaded a target class includes using a class loader that loaded the application class to look up the target class. The Klicnik reference neither discloses nor suggests this feature.

Claims 10 and 20 are additionally considered to distinguish over the art in reciting, responsive to a second application class loading a second plug-in class, the steps of identifying the target class loader within the class loader hierarchy that loaded the target class; identifying the plug-in class loader that is provided for and delegates to the target class loader; and loading the second plug-in class using the plug-in class loader. The Klicnik reference neither discloses nor suggests this feature.

XI. Conclusion

It is respectfully urged that the subject application is patentable over the cited Klicnik reference, and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number, if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: July 5, 2007

Respectfully submitted,

/James O. Skarsten/

James O. Skarsten
Reg. No. 28,346
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

Annotated Sheet

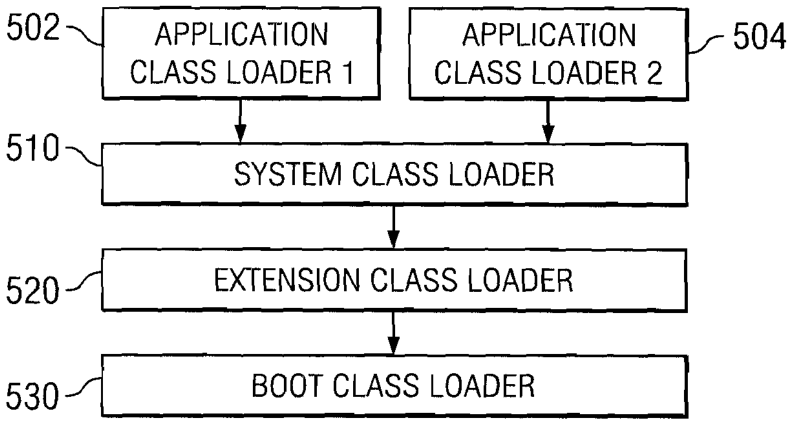


FIG. 5A
(Prior Art)

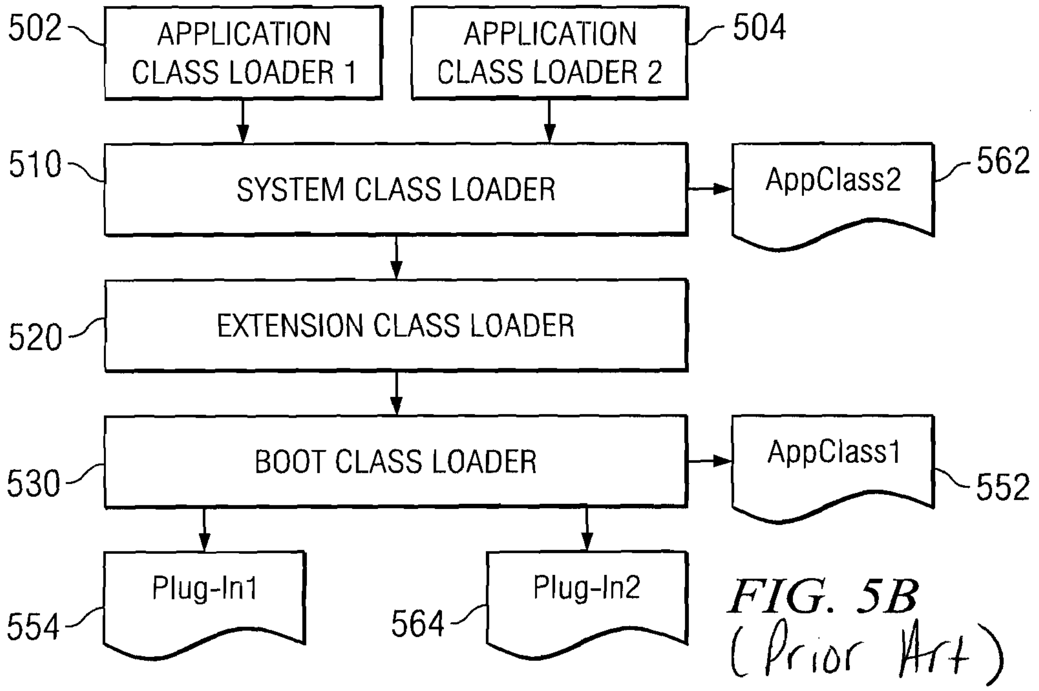


FIG. 5B
(Prior Art)

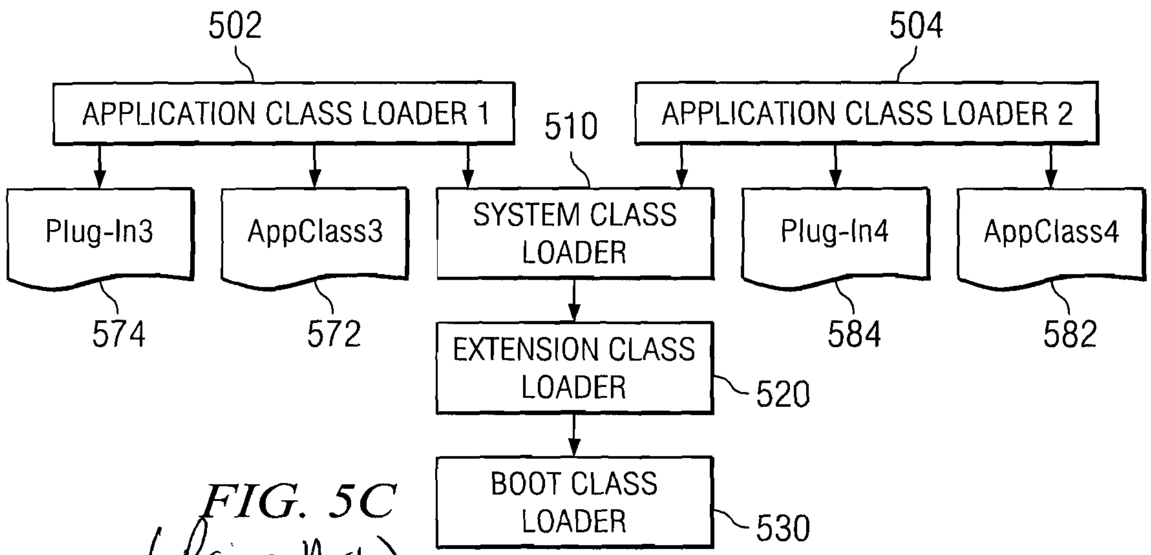


FIG. 5C
(Prior Art)